DiffDock: Diffusion Steps, Twists, and Turns for Molecular Docking

Gabriele Corso*, **Hannes Stärk***, **Bowen Jing***, **Regina Barzilay** & **Tommi Jaakkola**
CSAIL, Massachusetts Institute of Technology

Abstract

Predicting the binding structure of a small molecule ligand to a protein—a task known as *molecular docking*—is critical to drug design. Recent deep learning methods that treat docking as a regression problem have decreased runtime compared to traditional search-based methods but have yet to offer substantial improvements in accuracy. We instead frame molecular docking as a *generative* modeling problem and develop DIFFDOCK, a diffusion generative model over the non-Euclidean manifold of ligand poses. To do so, we map this manifold to the product space of the degrees of freedom (translational, rotational, and torsional) involved in docking and develop an efficient diffusion process on this space. Empirically, DIFFDOCK obtains a 38% top-1 success rate (RMSD<2Å) on PDBBind, significantly outperforming the previous state-of-the-art of traditional docking (23%) and deep learning (20%) methods. Moreover, DIFFDOCK has fast inference times and provides confidence estimates with high selective accuracy.

1 Introduction

Biological mechanisms are mediated by proteins, whose functions may be modulated by small molecule ligands (such as drugs) that bind to them. Thus, a crucial task in computational drug design is *molecular docking*—predicting the position, orientation, and conformation of a ligand when bound to a target protein—from which the effect of the ligand (if any) might be inferred. Recent works [1, 2] have developed deep learning models to predict the binding pose in one shot, framing docking as a regression problem. However, we argue (Appendix B) that common molecular docking applications and evaluation metrics better correspond to maximizing the likelihood of the data under a generative process. We thus frame molecular docking as a *generative modeling problem*—given a ligand and target protein structure, we learn a distribution over ligand poses.

We develop *docking diffusion* (DIFFDOCK), a diffusion generative model (DGM) over the space of ligand poses for molecular docking. We define a diffusion process over the degrees of freedom involved in docking: the position of the ligand relative to the protein (locating the binding pocket), its orientation in the pocket, and the m torsion angles describing its conformation. While DGMs have been applied to other problems in molecular machine learning [3, 4, 5], existing approaches are ill-suited for molecular docking, where the space of ligand poses lies near an $(m+6)$-dimensional submanifold $\mathcal{M} \subset \mathbb{R}^{3n}$. To develop DIFFDOCK, we recognize that the docking degrees of freedom define $\mathcal{M}$ as the space of poses accessible via a set of allowed *ligand pose transformations*. We use this idea to map elements in $\mathcal{M}$ to the *product space* of the groups corresponding to those transformations, where a DGM can be developed and trained more efficiently. We then train a *confidence model* to rank the DGM samples and select the final prediction.

Empirically, on the standard blind docking benchmark PDBBind, DIFFDOCK achieves 37% of top-1 predictions with ligand root mean square distance (RMSD) below 2Å, nearly doubling performance

*Equal contribution. Correspondance to `{gcorso, hstark, bjing}@mit.edu`.

NeurIPS 2022 Workshop on Score-Based Methods.

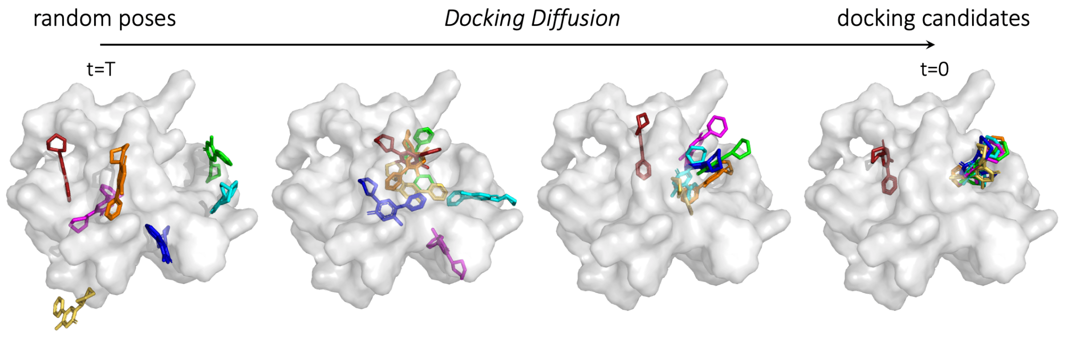

Figure 1: DiffDock's diffusion generative process. Initial poses are randomly sampled before the score model updates their location, rotation, and torsion angles in each denoising step to produce docking candidates.

of the previous state-of-the-art deep learning model (20%). DiffDock also significantly outperforms state-of-the-art classical search-based docking methods (23%), while still being significantly faster.

2 Method

A ligand pose is an assignment of atomic positions in $\mathbb{R}^3$, so in principle we can regard a pose $\mathbf{x}$ as an element in $\mathbb{R}^{3n}$, where n is the number of atoms. However, this encompasses far more degrees of freedom than are relevant in molecular docking. Traditional docking methods, as well as most ML ones, take as input a seed conformation $\mathbf{c} \in \mathbb{R}^{3n}$ of the ligand in isolation, and change only the relative position and the torsion degrees of freedom in the final bound conformation. The space of ligand poses consistent with $\mathbf{c}$ is, therefore, a $(m+6)$-dimensional submanifold $\mathcal{M}_\mathbf{c} \subset \mathbb{R}^{3n}$, where m is the number of rotatable bonds. We follow this paradigm of taking as input a seed conformation $\mathbf{c}$, and formulate molecular docking as learning a probability distribution $p_\mathbf{c}(\mathbf{x} \mid \mathbf{y})$ over the manifold $\mathcal{M}_\mathbf{c}$, conditioned on a protein structure $\mathbf{y}$.

DGMs on submanifolds have been formulated by [6] in terms of projecting a diffusion in ambient space onto the submanifold. However, the kernel $p(\mathbf{x}_t \mid \mathbf{x}_0)$ of such diffusion is not available in closed form and must be sampled numerically with a geodesic random walk, making training very inefficient. We instead define a one-to-one mapping to another, "nicer" manifold where the diffusion kernel can be sampled directly, and develop a DGM in that manifold. To start, we restate the discussion in the last paragraph as follows: *Any ligand pose consistent with a seed conformation be reached by a combination of (1) ligand translations (2) ligand rotations and (3) changes to torsion angles.* This suggests that given a continuous family of ligand pose transformations corresponding to the $m+6$ degrees of freedom, a distribution on $\mathcal{M}_\mathbf{c}$ can be lifted to a distribution on the product manifold of the corresponding groups.

2.1 Ligand Pose Transformations

We associate translations of ligand position with the 3D translation group $\mathbb{T}(3)$, rigid rotations of the ligand with the 3D rotation group $SO(3)$, and changes in torsion angles at each rotatable bond with a copy of the 2D rotation group $SO(2)$. More formally, we define operations of each of these groups on a ligand pose $\mathbf{c} \in \mathbb{R}^{3n}$. The translation $A_{\text{tr}} : \mathbb{T}(3) \times \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ is defined straightforwardly as $A_{\text{tr}}(\mathbf{r}, \mathbf{x})_i = \mathbf{x}_i + \mathbf{r}$ using the isomorphism $\mathbb{T}(3) \cong \mathbb{R}^3$ and where $\mathbf{x}_i \in \mathbb{R}^3$ is the position of the ith atom. Similarly, the rotation $A_{\text{rot}} : SO(3) \times \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ is defined by $A_{\text{rot}}(R, \mathbf{x})_i = R(\mathbf{x}_i - \bar{\mathbf{x}}) + \bar{\mathbf{x}}$ where $\bar{\mathbf{x}} = \frac{1}{n}\sum \mathbf{x}_i$, corresponding to rotations around the (unweighted) center of mass of the ligand.

Many valid definitions of a change in torsion angles are possible, as the torsion angle around any bond (a_i, b_i) can be updated by rotating the a_i side, the b_i side, or both. However, we can *disentangle* changes in torsion angles from rotations or translations by defining them to cause a minimal perturbation (in an RMSD sense) to the structure:

Definition. *Let $B_{k,\theta_k}(\mathbf{x}) \in \mathbb{R}^{3n}$ be any valid torsion update by θ_k around the kth rotatable bond (a_k, b_k). We define $A_{tor} : SO(2)^m \times \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ such that*

$$A_{tor}(\boldsymbol{\theta}, \mathbf{x}) = \mathrm{RMSDAlign}(\mathbf{x}, (B_{1,\theta_1} \circ \cdots B_{m,\theta_m})(\mathbf{x})) \tag{1}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots \theta_m)$ *and* $\mathrm{RMSDAlign}(\mathbf{x}, \mathbf{x}') = \arg\min_{\mathbf{x}^\dagger \in \{g\mathbf{x}' | g \in SE(3)\}} \mathrm{RMSD}(\mathbf{x}, \mathbf{x}^\dagger)$

This means that we apply all the m torsion updates in any manner, and then perform a global RMSD alignment with the unmodified pose. The definition is motivated by ensuring that the infinitesimal effect of a torsion is *orthogonal* to any rototranslation, which has a pleasing physical interpretation as inducing no *linear or angular momentum* (Appendix A):

Proposition 1. *Let* $\mathbf{x}(t) := A_{tor}(t\boldsymbol{\theta}, \mathbf{x})$ *for some* $\boldsymbol{\theta}$ *and where* $t\boldsymbol{\theta} = (t\theta_1, \ldots t\theta_m)$*. Then the linear and angular momentum are zero:* $\frac{d}{dt}\bar{\mathbf{x}}|_{t=0} = 0$ *and* $\sum_i (\mathbf{x}_i - \bar{\mathbf{x}}) \times \frac{d}{dt}\mathbf{x}_i|_{t=0} = 0$ *where* $\bar{\mathbf{x}} = \frac{1}{n}\sum_i \mathbf{x}$*.*

Now consider the product space $\mathbb{P} = \mathbb{T}^3 \times SO(3) \times SO(2)^m$ and define $A : \mathbb{P} \times \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ as

$$A((\mathbf{r}, R, \boldsymbol{\theta}), \mathbf{x}) = A_{\text{tr}}(\mathbf{r}, A_{\text{rot}}(R, A_{\text{tor}}(\boldsymbol{\theta}, \mathbf{x}))) \tag{2}$$

These definitions collectively provide the sought-after product space corresponding to the docking degrees of freedom. Indeed, for a seed ligand conformation $\mathbf{c}$, we can formally define the space of ligand poses $\mathcal{M}_\mathbf{c} = \{A(g, \mathbf{c}) \mid g \in \mathbb{P}\}$, which corresponds precisely to the intuitive notion of the space of ligand poses that can be reached by rigid-body motion plus torsion angle flexibility.

2.2 Diffusion on the Product Space

We now proceed to show how the product space can be used to learn a DGM over ligand poses in $\mathcal{M}_s$. First, we need a theoretical result (proof in Appendix A):

Proposition 2. *For a given seed conformation* $\mathbf{c}$*, the map* $A(\cdot, \mathbf{c}) : \mathbb{P} \to \mathcal{M}_\mathbf{c}$ *is a bijection.*

which means that the inverse $A_\mathbf{c}^{-1} : \mathcal{M}_\mathbf{c} \to \mathbb{P}$ given by $A(g, \mathbf{c}) \mapsto g$ maps ligand poses $\mathbf{x} \in \mathcal{M}_\mathbf{c}$ to points on the product space $\mathbb{P}$. We are now ready to develop a diffusion process on $\mathbb{P}$.

[6] established that the DGM framework transfers straightforwardly to Riemannian manifolds. Thus, to implement a diffusion model on $\mathbb{P}$, it suffices to develop a method for sampling from and computing the score of the diffusion kernel on $\mathbb{P}$. Since $\mathbb{P}$ is a product manifold, the forward diffusion proceeds independently in each manifold [7], and the tangent space is a direct sum: $T_g\mathbb{P} = T_\mathbf{r}\mathbb{T}_3 \oplus T_R SO(3) \oplus T_{\boldsymbol{\theta}} SO(2)^m \cong \mathbb{R}^3 \oplus \mathbb{R}^3 \oplus \mathbb{R}^m$ where $g = (\mathbf{r}, R, \boldsymbol{\theta})$. Thus, it suffices to sample from the diffusion kernel and regress against its score in each group independently.

In all three groups, we define the forward SDE as $d\mathbf{x} = \sqrt{d\sigma^2(t)/dt}\, d\mathbf{w}$ where $\sigma^2 = \sigma^2_{\text{tr}}$, σ^2_{rot}, or σ^2_{tor} for $\mathbb{T}(3)$, $SO(3)$, and $SO(2)^m$ respectively and where $\mathbf{w}$ is the corresponding Brownian motion. Since $\mathbb{T}(3) \cong \mathbb{R}^3$, the translational case is trivial and involves sampling and computing the score of a standard Gaussian with variance $\sigma^2(t)$. The diffusion kernel on $SO(3)$ is given by the $IGSO(3)$ distribution [8, 9], which can be sampled in the axis-angle parameterization by sampling a unit vector $\hat{\boldsymbol{\omega}} \in \mathfrak{so}(3)$ uniformly and random angle $\omega \in [0, \pi]$ according to

$$p(\omega) = \frac{1 - \cos\omega}{\pi} f(\omega) \quad \text{where} \quad f(\omega) = \sum_{l=0}^{\infty} (2l+1) \exp(-l(l+1)\sigma^2) \frac{\sin((l+1/2)\omega)}{\sin(\omega/2)} \tag{3}$$

Further, the score of the diffusion kernel is $\nabla \ln p_t(R' \mid R) = (\frac{d}{d\omega} \log f(\omega))\hat{\omega} \in T_{R'} SO(3)$, where $R' = \mathbf{R}(\omega\hat{\omega})R$ is the result of applying Euler vector $\omega\hat{\boldsymbol{\omega}}$ to R. The score computation and sampling can be accomplished efficiently by precomputing the truncated infinite series and interpolating the CDF of $p(\omega)$, respectively. Finally, the $SO(2)^m$ group is diffeomorphic to the torus $\mathbb{T}^m$, on which the diffusion kernel is a *wrapped normal distribution* with variance $\sigma^2(t)$. This can be sampled directly and the score can be precomputed as a truncated infinite series [5].

2.3 Training, Inference and Architecture

Diffusion model. Although we have defined the diffusion kernel and score matching objectives on $\mathbb{P}$, we nevertheless develop the training and inference procedures to operate on ligand poses in 3D coordinates directly. Providing the full 3D structure, to the score model allows it to reason about physical interactions using $SE(3)$ equivariant models, not be dependent on arbitrary definitions of torsion angles [5] and better generalize to unseen complexes.

The training and inference procedures technically depend on the choice of seed conformation $\mathbf{c}$ used to define the mapping between $\mathcal{M}_\mathbf{c}$ and the product space. However, providing a definite choice of $\mathbf{c}$

to the score model introduces an arbitrary inference-time parameter that may affect the final predicted distribution, which is undesirable. Thus, we develop approximate training and inference procedures that remove the dependence on the $\mathbf{c}$; intuitively, these assume that updates to points in the product space $\mathbb{P}$ can be applied to ligand poses in $\mathcal{M}_\mathbf{c}$ directly. While these are only an approximation of the theoretically correct procedures, we find that they work well in practice (Appendix C).

Confidence model. Once the diffusion model is trained, in order to collect training data for the confidence model, we run the diffusion model to obtain a set of candidate poses for every training example and generate labels by testing whether or not each pose has RMSD below 2Å. The confidence model is then trained with cross-entropy loss to correctly predict the binary label for each pose. During inference, the diffusion model is run to generate N poses in parallel, which are passed to the confidence model that ranks them based on its confidence that they have an RMSD below 2Å.

Model Architecture We construct the score model $\mathbf{s}(\mathbf{x}, \mathbf{y}, t)$ and the confidence model $\mathbf{d}(\mathbf{x}, \mathbf{y})$ to take as input the current ligand pose $\mathbf{x}$ and protein structure $\mathbf{y}$ in 3D space. The score model and confidence model have similar architectures based on $SE(3)$-equivariant convolutional networks. The output of the score model must be in the tangent space $T_\mathbf{r}\mathbb{T}_3 \oplus T_R SO(3) \oplus T_{\boldsymbol{\theta}} SO(2)^m$. The space $T_\mathbf{r}\mathbb{T}_3 \cong \mathbb{R}^3$ corresponds to translation vectors and $T_R SO(3) \cong \mathbb{R}^3$ to rotation (Euler) vectors, both of which are $SE(3)$-equivariant. Finally, $T_{\boldsymbol{\theta}} SO(2)^m$ corresponds to scores on $SE(3)$-invariant quantities. The architectural components are detailed in Appendix D.

3 Experiments

Table 1: **PDBBind blind docking.** Percentage of predictions with RMSD < 2Å and the median RMSD. For our method in parenthesis we specify the number of sampled poses.

	Top-1 RMSD (Å)		Top-5 RMSD (Å)		Average
Method	%<2	Med.	%<2	Med.	Runtime (s)
QVinaW [10]	20.9	7.7			49*
GNINA [11]	22.9	7.7	32.9	4.5	127
SMINA [12]	18.7	7.1	29.3	4.6	126*
GLIDE [13]	21.8	9.3			1405*
EquiBind [1]	5.5	6.2	-	-	**0.04**
TANKBind [2]	20.4	4.0	24.5	3.4	0.7/2.5
DiffDock (10)	34.5±0.3	3.61±0.05	40.1±1.1	2.79±0.09	10
DiffDock (40)	**37.5±0.5**	**3.46±0.04**	**43.3±0.5**	**2.53±0.05**	40

Experimental setup. We evaluate our method on the complexes from PDBBind [14], a large collection of protein-ligand structures collected from PDB [15], which was used with time-based splits to benchmark many previous works [1, 16, 2]. We compare DiffDock with state-of-the-art scoring methods SMINA [12], QuickVina-W [10], GLIDE [13], and GNINA [11] and the recent deep learning methods EquiBind and TANKBind presented above. Extensive details about the experimental setup, data, baselines, and implementation are in Appendix E.2.

Docking accuracy. As the results of Table 1 show, DiffDock significantly outperforms all previous methods. In particular, DiffDock obtains an impressive 36.6% top-1 RMSD below 2Å,vastly surpassing the performance of state-of-the-art commercial software such as GLIDE (21.8%) and the previous state-of-the-art deep learning method TANKBind (20.4%).

4 Conclusion

We presented DiffDock, a diffusion generative model tailored to the task of molecular docking, representing a paradigm shift from previous regression-based deep learning approaches. To produce a fast and accurate generative model, we designed a diffusion process over the manifold describing the main degrees of freedom of the task. Empirically, DiffDock outperforms the state-of-the-art by large margins on PDBBind. Thus, DiffDock can offer great value for many existing real-world pipelines and opens up new avenues of research on similar ideas in protein-protein docking.

References

[1] Hannes Stärk, Octavian Ganea, Lagnajit Pattanaik, Regina Barzilay, and Tommi Jaakkola. Equibind: Geometric deep learning for drug binding structure prediction. In *International Conference on Machine Learning*, pages 20503–20521. PMLR, 2022.

[2] Wei Lu, Qifeng Wu, Jixian Zhang, Jiahua Rao, Chengtao Li, and Shuangjia Zheng. Tankbind: Trigonometry-aware neural networks for drug-protein binding structure prediction. *Advances in neural information processing systems*, 2022.

[3] Minkai Xu, Lantao Yu, Yang Song, Chence Shi, Stefano Ermon, and Jian Tang. Geodiff: A geometric diffusion model for molecular conformation generation. In *International Conference on Learning Representations*, 2021.

[4] Emiel Hoogeboom, Victor Garcia Satorras, Clement Vignac, and Max Welling. Equivariant diffusion for molecule generation in 3d. In *International Conference on Machine Learning*, pages 8867–8887. PMLR, 2022.

[5] Bowen Jing, Gabriele Corso, Jeffrey Chang, Regina Barzilay, and Tommi Jaakkola. Torsional diffusion for molecular conformer generation. *arXiv preprint arXiv:2206.01729*, 2022.

[6] Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modeling. *arXiv preprint arXiv:2202.02763*, 2022.

[7] Emanuele Rodolà, Zorah Lähner, Alexander M Bronstein, Michael M Bronstein, and Justin Solomon. Functional maps representation on product manifolds. In *Computer Graphics Forum*, volume 38, pages 678–689. Wiley Online Library, 2019.

[8] Dmitry I Nikolayev and Tatjana I Savyolov. Normal distribution on the rotation group so (3). *Textures and Microstructures*, 29, 1970.

[9] Adam Leach, Sebastian M Schmon, Matteo T Degiacomi, and Chris G Willcocks. Denoising diffusion probabilistic models on so (3) for rotational alignment. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[10] Nafisa M. Hassan, Amr A. Alhossary, Yuguang Mu, and Chee-Keong Kwoh. Protein-ligand blind docking using quickvina-w with inter-process spatio-temporal integration. *Scientific Reports*, 7(1):15451, Nov 2017.

[11] Andrew T McNutt, Paul Francoeur, Rishal Aggarwal, Tomohide Masuda, Rocco Meli, Matthew Ragoza, Jocelyn Sunseri, and David Ryan Koes. Gnina 1.0: molecular docking with deep learning. *Journal of cheminformatics*, 13(1):1–20, 2021.

[12] David Ryan Koes, Matthew P Baumgartner, and Carlos J Camacho. Lessons learned in empirical scoring with smina from the csar 2011 benchmarking exercise. *Journal of chemical information and modeling*, 53(8):1893–1904, 2013.

[13] Thomas A Halgren, Robert B Murphy, Richard A Friesner, Hege S Beard, Leah L Frye, W Thomas Pollard, and Jay L Banks. Glide: a new approach for rapid, accurate docking and scoring. 2. enrichment factors in database screening. *Journal of medicinal chemistry*, 2004.

[14] Zhihai Liu, Minyi Su, Li Han, Jie Liu, Qifan Yang, Yan Li, and Renxiao Wang. Forging the basis for developing protein–ligand interaction scoring functions. *Accounts of Chemical Research*, 50(2):302–309, 2017.

[15] H. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide Protein Data Bank. *Nat Struct Biol*, 10(12):980, Dec 2003.

[16] Mikhail Volkov, Joseph-André Turk, Nicolas Drizard, Nicolas Martin, Brice Hoffmann, Yann Gaston-Mathé, and Didier Rognan. On the frustration to predict binding affinities from protein–ligand structures with deep neural networks. *Journal of Medicinal Chemistry*, 65(11):7946–7958, Jun 2022.

[17] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[18] Marco Necci, Damiano Piovesan, and Silvio CE Tosatto. Critical assessment of protein intrinsic disorder prediction. *Nature methods*, 18(5):472–481, 2021.

[19] Nathaniel Bennett, Brian Coventry, Inna Goreshnik, Buwei Huang, Aza Allen, Dionne Vafeados, Ying Po Peng, Justas Dauparas, Minkyung Baek, Lance Stewart, et al. Improving de novo protein binder design with deep learning. *bioRxiv*, 2022.

[20] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint*, 2018.

[21] Mario Geiger, Tess Smidt, Alby M., Benjamin Kurt Miller, Wouter Boomsma, Bradley Dice, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, Simon Batzner, Martin Uhrin, Jes Frellsen, Nuri Jung, Sophia Sanborn, Josh Rackers, and Michael Bailey. Euclidean neural networks: e3nn, 2020.

[22] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, and Alexander Rives. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *arXiv*, 2022.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[24] Kristof Schütt, Pieter-Jan Kindermans, Huziel Enoc Sauceda Felix, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in neural information processing systems*, 2017.

[25] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

[26] Rocco Meli and Philip C. Biggin. spyrmsd: symmetry-corrected rmsd calculations in python. *Journal of Cheminformatics*, 12(1):49, 2020.

[27] Amr Alhossary, Stephanus Daniel Handoko, Yuguang Mu, and Chee-Keong Kwoh. Fast, accurate, and reliable molecular docking with QuickVina 2. *Bioinformatics*, 31(13):2214–2216, 02 2015.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Radoslav Krivák and David Hoksza. P2rank: machine learning based tool for rapid and accurate prediction of ligand binding sites from protein structure. *Journal of cheminformatics*, 10(1):1–12, 2018.

[30] Srinivas Ramachandran, Pradeep Kota, Feng Ding, and Nikolay V Dokholyan. Automated minimization of steric clashes in protein structures. *Proteins*, 79(1):261–270, January 2011.

A Proofs

A.1 Proof of Proposition 1

Proposition 1. *Let* $\mathbf{x}(t) := A_{tor}(t\boldsymbol{\theta}, \mathbf{x})$ *for some* $\boldsymbol{\theta}$ *and where* $t\boldsymbol{\theta} = (t\theta_1, \dots t\theta_m)$*. Then the linear and angular momentum are zero:* $\frac{d}{dt}\bar{\mathbf{x}}|_{t=0} = 0$ *and* $\sum_i (\mathbf{x}_i - \bar{\mathbf{x}}) \times \frac{d}{dt}\mathbf{x}_i|_{t=0} = 0$ *where* $\bar{\mathbf{x}} = \frac{1}{n}\sum_i \mathbf{x}$*.*

Proof. Let $\mathbf{x}(t) = R(t)(B(t\boldsymbol{\theta}, \mathbf{x}) - \bar{\mathbf{x}}_0) + \bar{\mathbf{x}}_0 + \mathbf{p}(t)$ where $B(t\boldsymbol{\theta}, \cdot) = B_{1,t\theta_1} \circ \cdots B_{m,t\theta_m}$ and $R(t), \mathbf{p}(t)$ are the rotation (around $\bar{\mathbf{x}}_0 := \bar{\mathbf{x}}(0)$) and translation associated with the optimal alignment between $B(t\boldsymbol{\theta}, \mathbf{x})$ and $\mathbf{x}$. By definition of RMSD, we have

$$||\mathbf{x}(t) - \mathbf{x}(0)|| = \min_{R,\mathbf{p}} ||R(t)(B(t\boldsymbol{\theta}, \mathbf{x}(0)) - \bar{\mathbf{x}}_0) + \bar{\mathbf{x}}_0 + \mathbf{p}(t) - \mathbf{x}(0)|| \tag{4}$$

which, in the limit of $t \to 0$, becomes

$$\left|\left| \frac{d}{dt}\mathbf{x}(t) \right|\right|^2_{t=0} = \min_{R,\mathbf{p}} \left|\left| \frac{d}{dt} \left(R(t)(B(t\boldsymbol{\theta}, \mathbf{x}(0)) - \bar{\mathbf{x}}_0) + \mathbf{p}(t)\right) \right|\right|^2_{t=0} \tag{5}$$

The derivative in the LHS of Equation 5 at $t = 0$ is

$$R'(t)(\mathbf{x} - \bar{\mathbf{x}}) + B'(t\boldsymbol{\theta}, \mathbf{x}(0)) + \mathbf{p}'(t) \tag{6}$$

and represents the instantaneous velocity of the points $\mathbf{x}_i$ at $t = 0$. Denoting $\mathbf{r}_i = \mathbf{x}_i - \bar{\mathbf{x}}_0$, our objective is to minimize $\sum_i ||\mathbf{r}'_i||^2$. Then, if we describe $R'(t)(\mathbf{x} - \bar{\mathbf{x}}_0) = R'(t)\mathbf{r}$ by an angular velocity $\boldsymbol{\omega}$ and abbreviate $B'(t\boldsymbol{\theta}, \mathbf{x}(0))_i := \mathbf{b}_i$ and $\mathbf{p}' = \mathbf{v}$, we have

$$\begin{aligned}
\sum_i ||\mathbf{r}'_i||^2 &= \sum_i (\mathbf{b}_i + \boldsymbol{\omega} \times \mathbf{r}_i + \mathbf{v}) \cdot (\mathbf{b}_i + \boldsymbol{\omega} \times \mathbf{r}_i + \mathbf{v}) \\
&= \sum_i \left[||\mathbf{b}_i||^2 + 2\mathbf{b}_i \cdot (\boldsymbol{\omega} \times \mathbf{r}_i) + 2\mathbf{b}_i \cdot \mathbf{v} + (\boldsymbol{\omega} \times \mathbf{r}_i) \cdot (\boldsymbol{\omega} \times \mathbf{r}_i) + 2(\boldsymbol{\omega} \times \mathbf{r}_i) \cdot \mathbf{v} + ||\mathbf{v}||^2\right] \\
&= \sum_i ||\mathbf{b}_i||^2 + 2\boldsymbol{\omega} \cdot \sum_i (\mathbf{r}_i \times \mathbf{b}_i) + 2\left(\sum_i \mathbf{b}_i\right) \cdot \mathbf{v} + n\,||\mathbf{v}||^2 + \boldsymbol{\omega}^T \mathcal{I}(\mathbf{r})\boldsymbol{\omega}
\end{aligned} \tag{7}$$

where we have used the fact that $\sum_i \mathbf{r}_i = 0$ and where $\mathcal{I}(r) = \left(\sum_i \mathbf{r}_i \cdot \mathbf{r}_i\right) I - \sum_i \mathbf{r}_i\mathbf{r}_i^T$ is the 3×3 *inertia tensor*. Then setting gradients with respect to $\mathbf{v}, \boldsymbol{\omega}$ gives

$$\mathbf{v} = -\frac{1}{n}\sum_i \mathbf{b}_i \quad \text{and} \quad \boldsymbol{\omega} = -\mathcal{I}(\mathbf{r})^{-1} \left(\sum_i \mathbf{r}_i \times \mathbf{b}_i\right) \tag{8}$$

Now with $\mathbf{r}'_i = \mathbf{b}_i + \boldsymbol{\omega} \times \mathbf{r}_i + \mathbf{v}$ we evaluate the linear momentum

$$\frac{1}{n}\sum_i \mathbf{r}'_i = \frac{1}{n}\left(\sum_i \mathbf{b}_i + \boldsymbol{\omega} \times \sum_i \mathbf{r}_i + n\mathbf{v}\right) = 0 \tag{9}$$

Similarly, we evaluate the angular momentum

$$\begin{aligned}
\sum_i \mathbf{r}_i \times \mathbf{r}'_i &= \sum_i \mathbf{r}_i \times \mathbf{b}_i + \sum_i \mathbf{r}_i \times (\boldsymbol{\omega} \times \mathbf{r}_i) + \sum_i \mathbf{r}_i \times \mathbf{v} \\
&= \sum_i \mathbf{r}_i \times \mathbf{b}_i + \mathcal{I}(\mathbf{r})\boldsymbol{\omega} = 0
\end{aligned} \tag{10}$$

Thus, the linear and angular momentum are zero at $t = 0$ for arbitrary $\mathbf{x}(0)$. □

Note that since we did not use the particular form of $B(t\boldsymbol{\theta}, \mathbf{x})$ in the above proof, we have shown that RMSD alignment can be used to disentangle rotations and translations from the infinitesimal action of any arbitrary function.

A.2 Proof of Proposition 2

Proposition 2. *For a given seed conformation* $\mathbf{c}$*, the map* $A(\cdot, \mathbf{c}) : \mathbb{P} \to \mathcal{M}_\mathbf{c}$ *is a bijection.*

Proof. Since we defined $\mathcal{M}_\mathbf{c} = \{A(g, \mathbf{c}) \mid g \in \mathbb{P}\}$, $A(\cdot, \mathbf{c})$ is automatically surjective. We now show that it is injective. Assume for the sake of contradiction that $A(\cdot, \mathbf{c})$ is not injective, so that there exist elements of the product space $g_1, g_2 \in \mathbb{P}$ with $g_1 \neq g_2$ but with $A(g_1, \mathbf{c}) = A(g_2, \mathbf{c}) = \mathbf{c}'$. That is,

$$A_{\text{tr}}(\mathbf{r}_1, A_{\text{rot}}(R_1, A_{\text{tor}}(\boldsymbol{\theta}_1, \mathbf{c}))) = A_{\text{tr}}(\mathbf{r}_2, A_{\text{rot}}(R_2, A_{\text{tor}}(\boldsymbol{\theta}_2, \mathbf{c}))) \tag{11}$$

which we abbreviate as $\mathbf{c}^{(1)} = \mathbf{c}^{(2)}$. Since only A_{tr} changes the center of mass $\sum_i \mathbf{c}_i/n$, we have $\sum_i \mathbf{c}_i^{(1)}/n = \sum_i \mathbf{c}_i/n + \mathbf{r}_1$ and $\sum_i \mathbf{c}_i^{(2)}/n = \sum_i \mathbf{c}_i/n + \mathbf{r}_2$. However, since $\mathbf{c}^{(1)} = \mathbf{c}^{(2)}$, this implies $\mathbf{r}_1 = \mathbf{r}_2$. Next, consider the torsion angles $\boldsymbol{\tau}_1 = (\tau_1^{(1)}, \dots \tau_m^{(1)})$ of $\mathbf{c}^{(1)}$ corresponding to some choice of dihedral angles at each rotatable bond. Because A_{tr} and A_{rot} are rigid-body motions, only A_{tor} changes the dihedral angles; in particular, by definition we have $\tau_i^{(1)} \cong \tau_i + \theta_i^{(1)} \mod 2\pi$ and $\tau_i^{(2)} \cong \tau_i + \theta_i^{(2)} \mod 2\pi$ for all $i = 1, \dots m$. However, because $\tau_i^{(1)} = \tau_i^{(2)}$, this means $\theta_i^{(1)} \cong \theta_i^{(2)}$ for all i and therefore $\boldsymbol{\theta}_1 = \boldsymbol{\theta}_2$ (as elements of $SO(2)^m$). Now denote $\mathbf{c}^\star = A_{\text{tor}}(\boldsymbol{\theta}_1, \mathbf{c}) = A_{\text{tor}}(\boldsymbol{\theta}_2, \mathbf{c})$ and apply $A_{\text{tr}}(-\mathbf{r}_1, \cdot) = A_{\text{tr}}(-\mathbf{r}_2, \cdot)$ to both sides of Equation 11. We then have

$$A_{\text{rot}}(R_1, \mathbf{c}^\star) = A_{\text{rot}}(R_2, \mathbf{c}^\star) \tag{12}$$

which further leads to

$$\mathbf{c}^\star - \bar{\mathbf{c}}^\star = R_1^{-1} R_2 (\mathbf{c}^\star - \bar{\mathbf{c}}^\star) \tag{13}$$

In general, this does not imply that $R_1 = R_2$. However, $R_1 \neq R_2$ is possible only if $\mathbf{c}^\star$ is *degenerate*, in the sense that all points are collinear along the shared axis of rotation of R_1, R_2. However, in practice conformers never consist of a collinear set of points, so we can safely assume $R_1 = R_2$. We now have $(\mathbf{r}_1, R_1, \boldsymbol{\theta}_1) = (\mathbf{r}_2, R_2, \boldsymbol{\theta}_2)$, or $g_1 = g_2$, contradicting our initial assumption. We thus conclude that $A(\cdot, \mathbf{c})$ is injective, completing the proof. □

B Docking as Generative Modeling

Although EquiBind and other ML methods have provided strong runtime improvements by avoiding an expensive optimization process over ligand poses, their performance has not reached yet that of traditional methods. As our analysis below argues, this may be caused by the models' uncertainty and the optimization of an objective function that does not correspond with how molecular docking is used and evaluated in practice.

Molecular docking objective. Molecular docking plays a critical role in drug discovery because the prediction of the 3D structure of a bound protein-ligand complex enables further computational and human expert analyses on the strength and properties of the binding interaction. Therefore, a docked prediction is only useful if its deviation from the true structure does not significantly affect the output of such analysis. Concretely, a prediction is considered acceptable when the distance between the structures (measured in terms of ligand RMSD) is below some small tolerance on the order of the length scale of atomic interactions (a few angstroms). Consequently, the standard evaluation metric used in the field has been the percentage of predictions with a ligand RMSD (relative to the ground truth) within some value ϵ.

However, the objective of maximizing the proportion of predictions with RMSD within some tolerance ϵ is not differentiable and therefore cannot be used for the training of models with stochastic gradient descent. Instead, we observe that maximizing the expected proportion of predictions with RMSD $< \epsilon$ corresponds to maximizing the likelihood of the true structure under the model's output distribution, in the limit as ϵ goes to 0.

Generative model. This observation motivates training a generative model to minimize an upper bound on the negative log-likelihood of the observed structures under the model's distribution. Thus, we view molecular docking as the problem of learning a distribution over ligand poses conditioned on the protein structure and develop a diffusion generative model over this space.

Confidence model. With a trained diffusion model, it is possible to sample an arbitrary number of ligand poses from the posterior distribution of poses according to the model. However, researchers are often interested in seeing only one or a small number of predicted poses for downstream analysis and an associated confidence measure[2]. Thus, we train a confidence model over the poses sampled by the diffusion model and ranks them based on its confidence they are within the error tolerance. The top-ranked ligand pose and the associated confidence are then taken as DIFFDOCK's top-1 prediction and the confidence score.

Problem with regression methods. The difficulty with the development of deep learning models for molecular docking lies in the intrinsic aleatoric uncertainty on the pose often present and the

[2]For example, the pLDDT confidence score of AlphaFold2 [17] has had a very significant impact in many applications [18, 19].

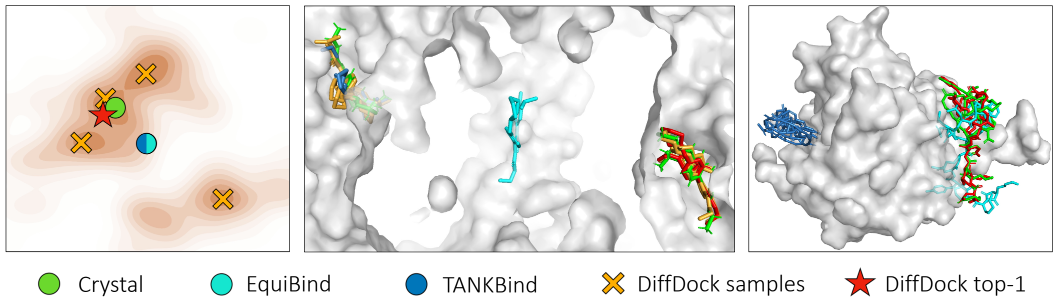

Figure 2: *Left:* Visual diagram of the advantage of generative models over regression models. Given uncertainty in the correct pose (represented by the orange distribution), regression models tend to predict the mean of the distribution, which may lie in a region of low density. *Center:* when there is a global symmetry in the protein (aleatoric uncertainty), EquiBind places the molecule in the center while DiffDock is able to sample all the true poses. *Right:* even in absence of strong aleatoric uncertainty, the epistemic uncertainty causes EquiBind's prediction to have steric clashes and TankBind's to have many self-intersections. DiffDock top-1 indicates the sample with the highest confidence, while DiffDock samples the other diffusion model samples.

complexity of the task compared with the limited model capacity and data available (epistemic uncertainty). Therefore, given the available co-variate information (only protein structure and ligand identity), any method will be unsure how to favor the correct among many viable alternative poses. Any regression-style method that is forced to select a single configuration that minimizes the expected square error would learn to predict the (weighted) mean of such alternatives. In contrast, a generative model with the same co-variate information would instead aim to capture the distribution over the alternatives, populating all/most of the significant modes even if similarly unable to distinguish the correct target. This behavior, illustrated in Figure 2, causes the regression-based models to produce significantly more physically implausible poses than our method. In particular, we observe frequent steric clashes (e.g., 26% of EquiBind's predictions) and self-intersections in EquiBind's and TANKBind's predictions (Figures 3 and 6). We found no intersections in DiffDock's predictions. Visualizations and quantitative evidence of these phenomena are in Appendix F.1.

C Training and Inference

In this section we present the training and inference procedures of the diffusion generative model. First, however, there are a few subtleties of the generative approach to molecular docking that are worth mentioning. Unlike the standard generative modeling setting where the dataset consists of many samples drawn from the data distribution, each training example $(\mathbf{x}^\star, \mathbf{y})$ of protein structure $\mathbf{y}$ and ground-truth ligand pose $\mathbf{x}^\star$ is the *only sample* from the corresponding conditional distribution $p_{\mathbf{x}^\star}(\cdot \mid \mathbf{y})$ defined over $\mathcal{M}_{\mathbf{x}^\star}$. Thus, the innermost training loop iterates over distinct conditional distributions $p_{\mathbf{x}^\star}(\cdot \mid \mathbf{y})$, along with a single sample from that distribution, rather than over samples from a common data distribution $p_{\text{data}}(\mathbf{x})$.

As discussed in Section 2, during inference $\mathbf{c}$ is the ligand structure generated with a method such as RDKit. However, during training we require $\mathcal{M}_\mathbf{c} = \mathcal{M}_{\mathbf{x}^\star}$ in order to define a bijection between $\mathbf{c} \in \mathcal{M}_{\mathbf{x}^\star}$ and $\mathbb{P}$. If we take $\mathbf{c} \in \mathcal{M}_{\mathbf{x}^\star}$, there will be a distribution shift between the manifolds $\mathcal{M}_\mathbf{c}$ considered at training time, and those considered at inference time. To circumvent this issue, at training time we predict $\mathbf{c}$ with RDKit and replace $\mathbf{x}^\star$ with $\arg\min_{\mathbf{x}^\dagger \in \mathcal{M}_\mathbf{c}} \text{RMSD}(\mathbf{x}^\star, \mathbf{x}^\dagger)$ using the conformer matching procedure described in [5].

The above paragraph may be rephrased more intuitively as follows: during inference, the generative model docks a ligand structure generated by RDKit, keeping its non-torsional degrees of freedom (e.g., local structures) fixed. At training time, however, if we train the score model with the local structures of the ground truth pose, this will not correspond to the local structures seen at inference time. Thus, at training time we replace the ground truth pose by generating a ligand structure with RDKit and aligning it to the ground truth pose while keeping the local structures fixed.

With these preliminaries, we now continue to the full procedures. The training and inference procedures of a score-based diffusion generative model on a Riemannian manifold consist of (1) sampling and regressing against the score of the diffusion kernel during training; and (2) sampling a geodesic random walk with the score as a drift term during inference [6]. Because we have developed the diffusion process on $\mathbb{P}$ but continue to provide the score model with elements in $\mathcal{M}_{\mathbf{c}} \subset \mathbb{R}^{3n}$, the full training and inference procedures involve repeatedly interconverting between the two spaces using the bijection given by the seed conformation $\mathbf{c}$.

Algorithm 1: Training procedure (single epoch)

```
Input: Training pairs $\{(\mathbf{x}^\star, \mathbf{y})\}$, RDKit predictions $\{\mathbf{c}\}$
foreach $\mathbf{c}, \mathbf{x}^\star, \mathbf{y}$ do
    Let $\mathbf{x}_0 \leftarrow \arg\min_{\mathbf{x}^\dagger \in \mathcal{M}_{\mathbf{c}}} \mathrm{RMSD}(\mathbf{x}^\star, \mathbf{x}^\dagger)$;
    Compute $(\mathbf{r}_0, R_0, \boldsymbol{\theta}_0) \leftarrow A_{\mathbf{c}}^{-1}(\mathbf{x}_0)$;
    Sample $t \sim \mathrm{Uni}([0,1])$;
    Sample $\Delta\mathbf{r}, \Delta R, \Delta\boldsymbol{\theta}$ from diffusion kernels $p_t^{\text{tr}}(\cdot \mid 0), p_t^{\text{rot}}(\cdot \mid 0), p_t^{\text{tor}}(\cdot \mid 0)$;
    Set $\mathbf{r}_t \leftarrow \mathbf{r}_0 + \Delta\mathbf{r}$;
    Set $R_t \leftarrow (\Delta R) R_0$;
    Set $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta} \mod 2\pi$;
    Compute $\mathbf{x}_t \leftarrow A((\mathbf{r}_t, R_t, \boldsymbol{\theta}_t), \mathbf{c})$;
    Predict scores $\alpha \in \mathbb{R}^3, \beta \leftarrow \mathbb{R}^3, \gamma \in \mathbb{R}^m = \mathbf{s}(\mathbf{x}_t, \mathbf{c}, \mathbf{y}, t)$ ;
    Take optimization step on loss
    $\mathcal{L} = ||\alpha - \nabla p_t^{\text{tr}}(\Delta\mathbf{r} \mid 0)||^2 + ||\beta - \nabla p_t^{\text{rot}}(\Delta R \mid 0)||^2 + ||\gamma - \nabla p_t^{\text{tor}}(\Delta\boldsymbol{\theta} \mid 0)||^2$
```

Algorithm 2: Inference procedure

```
Input: RDKit prediction $\mathbf{c}$, protein structure $\mathbf{y}$ (both centered at origin)
Output: Sampled ligand pose $\mathbf{x}_0$
Sample $\boldsymbol{\theta}_N \sim \mathrm{Uni}(SO(2)^m)$, $R_N \sim \mathrm{Uni}(SO(3))$, $\mathbf{r}_N \sim \mathcal{N}(0, \sigma^2_{\text{tor}}(T))$;
Let $\mathbf{x}_N = A((\mathbf{r}_N, R_N, \boldsymbol{\theta}_N), \mathbf{c})$;
for $n \leftarrow N$ to $1$ do
    Let $t = n/N$ and $\Delta\sigma^2_{\text{tr}} = \sigma^2_{\text{tr}}(n/N) - \sigma^2_{\text{tr}}((n-1)/N)$ and similarly for $\Delta\sigma^2_{\text{rot}}, \Delta\sigma^2_{\text{tor}}$;
    Predict scores $\alpha \in \mathbb{R}^3, \beta \in \mathbb{R}^3, \gamma \in \mathbb{R}^m \leftarrow \mathbf{s}(\mathbf{x}_n, \mathbf{c}, \mathbf{y}, t)$;
    Sample $\mathbf{z}_{\text{tr}}, \mathbf{z}_{\text{rot}}, \mathbf{z}_{\text{tor}}$ from $\mathcal{N}(0, \Delta\sigma^2_{\text{tr}}), \mathcal{N}(0, \Delta\sigma^2_{\text{rot}}), \mathcal{N}(0, \Delta\sigma^2_{\text{tor}})$ respectively;
    Set $\mathbf{r}_{n-1} \leftarrow \mathbf{r}_0 + \Delta\sigma^2_{\text{tr}}\alpha + \mathbf{z}_{\text{tr}}$;
    Set $R_{n-1} \leftarrow \mathbf{R}(\Delta\sigma^2_{\text{rot}}\beta + \mathbf{z}_{\text{rot}}) R_n)$;
    Set $\boldsymbol{\theta}_{n-1} \leftarrow \boldsymbol{\theta}_n + (\Delta\sigma^2_{\text{tor}}\gamma + \mathbf{z}_{\text{tor}}) \mod 2\pi$;
    Compute $\mathbf{x}_{n-1} \leftarrow A((\mathbf{r}_{n-1}, R_{n-1}, \boldsymbol{\theta}_{n-1}), \mathbf{c})$;
Return $\mathbf{x}_0$;
```

However, as noted in the main text, the dependence of these procedures on the exact choice of $\mathbf{c}$ is potentially problematic, as it suggests that at inference time, the model distribution may be different depending on the orientation and torsion angles of $\mathbf{c}$. Simply removing the dependence of the score model on $\mathbf{c}$ is not sufficient, since the update steps themselves still occur on $\mathbb{P}$ and require a choice of $\mathbf{c}$ to be mapped to $\mathcal{M}_{\mathbf{c}}$. However, notice that the update steps—in both training and inference—consist of (1) sampling the diffusion kernels at the origin; (2) applying these updates to the point on $\mathbb{P}$; and (3) transferring the point on $\mathbb{P}$ to $\mathcal{M}_{\mathbf{c}}$ via $A(\cdot, \mathbf{c})$. Might it instead be possible to apply the updates to 3D ligand poses $\mathbf{x} \in \mathcal{M}_{\mathbf{c}}$ *directly*?

It turns out that the notion of applying these steps to ligand poses "directly" corresponds to the formal notion of *group action*. The operations $A_{\text{tr}}, A_{\text{rot}}, A_{\text{tor}}$ that we have already defined are formally group actions if they satisfy $A_{(\cdot)}(g_1 g_2, \mathbf{x}) = A(g_1, A(g_2, \mathbf{x}))$. While true for $A_{\text{tr}}, A_{\text{rot}}$, this is not generally true for A_{tor} if we take $SO(2)^m$ to be the direct product group; however, the approximation is increasingly good as the magnitude of the torsion angle updates decreases. If we then define $\mathbb{P}$ to be the direct product group of its constituent groups, A is a group action of $\mathbb{P}$ on $\mathcal{M}_{\mathbf{c}}$, as the operations of $A_{\text{tr}}, A_{\text{rot}}, A_{\text{tor}}$ commute and are (under the approximation) individually group actions.

The implication of A being a group action can be seen as follows. Let $\delta = g_b g_a^{-1}$ be the update which brings $g_a \in \mathbb{P}$ to $g_b \in \mathbb{P}$ via left multiplication, and let $\mathbf{x}_a, \mathbf{x}_b$ be the corresponding ligand poses $A(g_a, \mathbf{c}), A(g_b, \mathbf{c})$. Then

$$\mathbf{x}_b = A(g_b g_a^{-1} g_a, \mathbf{c}) = A(\delta, \mathbf{x}_a) \tag{14}$$

which means that the updates δ can be applied directly to $\mathbf{x}_a$ using the operation A. The training and inference procedures then become Algorithm 3 and 4 below. The initial conformer $\mathbf{c}$ is no longer used, except in the the initial steps to define the manifold—to find the closest point to $\mathbf{x}^\star$ in training, and to sample $\mathbf{x}_N$ from the prior over $\mathcal{M}_\mathbf{c}$ in inference.

Conceptually speaking, this procedure corresponds to "forgetting" the location of the origin element on $\mathcal{M}_\mathbf{c}$, which is permissible because a change of the origin to some equivalent seed $\mathbf{c}' \in \mathcal{M}_\mathbf{c}$ merely translates—via right multiplication by $A_\mathbf{c}^{-1}(\mathbf{c}')$—the original and diffused data distributions on $\mathbb{P}$, but does not cause any changes on $\mathcal{M}_\mathbf{c}$ itself. The training and inference routines involve updates—formally left multiplications—to group elements, but as left multiplication on the group corresponds to group actions on $\mathcal{M}_\mathbf{c}$, the updates can act on $\mathcal{M}_\mathbf{c}$ directly, without referencing the origin $\mathbf{c}$.

We find that the approximation of A as a group action works quite well in practice, and use Algorithms 3 and 4 for all training and experiments discussed in the paper. Of course, disentangling the torsion updates from rotations in a way that makes A_{tor} exactly a group action would justify the procedure further, and we regard this as a possible direction of future work.

Algorithm 3: Approximate training procedure (single epoch)

```
Input: Training pairs $\{(\mathbf{x}^\star, \mathbf{y})\}$, RDKit predictions $\{\mathbf{c}\}$
foreach $\mathbf{c}, \mathbf{x}^\star, \mathbf{y}$ do
    Let $\mathbf{x}_0 \leftarrow \arg\min_{\mathbf{x}^\dagger \in \mathcal{M}_\mathbf{c}} \mathrm{RMSD}(\mathbf{x}^\star, \mathbf{x}^\dagger)$;
    Sample $t \sim \mathrm{Uni}([0,1])$;
    Sample $\Delta\mathbf{r}, \Delta R, \Delta\boldsymbol{\theta}$ from diffusion kernels $p_t^{\text{tr}}(\cdot \mid 0), p_t^{\text{rot}}(\cdot \mid 0), p_t^{\text{tor}}(\cdot \mid 0)$;
    Compute $\mathbf{x}_t \leftarrow A((\Delta\mathbf{r}, \Delta R, \Delta\boldsymbol{\theta}), \mathbf{x}_0)$;
    Predict scores $\alpha \in \mathbb{R}^3, \beta \leftarrow \mathbb{R}^3, \gamma \in \mathbb{R}^m = \mathbf{s}(\mathbf{x}_t, \mathbf{y}, t)$ ;
    Take optimization step on loss
      $\mathcal{L} = ||\alpha - \nabla p_t^{\text{tr}}(\Delta\mathbf{r} \mid 0)||^2 + ||\beta - \nabla p_t^{\text{rot}}(\Delta R \mid 0)||^2 + ||\gamma - \nabla p_t^{\text{tor}}(\Delta\boldsymbol{\theta} \mid 0)||^2$
```

Algorithm 4: Approximate inference procedure

```
Input: RDKit prediction $\mathbf{c}$, protein structure $\mathbf{y}$ (both centered at origin)
Output: Sampled ligand pose $\mathbf{x}_0$
Sample $\boldsymbol{\theta}_N \sim \mathrm{Uni}(SO(2)^m)$, $R_N \sim \mathrm{Uni}(SO(3))$, $\mathbf{r}_N \sim \mathcal{N}(0, \sigma^2_{\text{tor}}(T))$;
Let $\mathbf{x}_N = A((\mathbf{r}_N, R_N, \boldsymbol{\theta}_N), \mathbf{c})$;
for $n \leftarrow N$ to $1$ do
    Let $t = n/N$ and $\Delta\sigma^2_{\text{tr}} = \sigma^2_{\text{tr}}(n/N) - \sigma^2_{\text{tr}}((n-1)/N)$ and similarly for $\Delta\sigma^2_{\text{rot}}, \Delta\sigma^2_{\text{tor}}$;
    Predict scores $\alpha \in \mathbb{R}^3, \beta \in \mathbb{R}^3, \gamma \in \mathbb{R}^m \leftarrow \mathbf{s}(\mathbf{x}_n, \mathbf{y}, t)$;
    Sample $\mathbf{z}_{\text{tr}}, \mathbf{z}_{\text{rot}}, \mathbf{z}_{\text{tor}}$ from $\mathcal{N}(0, \Delta\sigma^2_{\text{tr}}), \mathcal{N}(0, \Delta\sigma^2_{\text{rot}}), \mathcal{N}(0, \Delta\sigma^2_{\text{tor}})$ respectively;
    Set $\Delta\mathbf{r} \leftarrow \mathbf{r}_0 + \Delta\sigma^2_{\text{tr}}\alpha + \mathbf{z}_{\text{tr}}$;
    Set $\Delta R \leftarrow \mathbf{R}(\Delta\sigma^2_{\text{rot}}\beta + \mathbf{z}_{\text{rot}})$;
    Set $\Delta\boldsymbol{\theta} \leftarrow \Delta\sigma^2_{\text{tor}}\gamma + \mathbf{z}_{\text{tor}}$;
    Compute $\mathbf{x}_{n-1} \leftarrow A((\Delta\mathbf{r}, \Delta R, \Delta\boldsymbol{\theta}), \mathbf{x}_n)$;
Return $\mathbf{x}_0$;
```

D Architecture Details

We use convolutional networks based on tensor products of irreducible representations (irreps) of SO(3) [20] as architecture for both the score and confidence models. In particular, these are implemented using the `e3nn` library [21]. Below, $\otimes_w$ refers to the spherical tensor product of irreps with path weights w and $\oplus$ refers to normal vector addition (with possibly padded inputs). Features have multiple channels for each irrep. Both the architectures can be decomposed in three main parts: embedding layer, interaction layers and output layer. We outline each of them below.

D.1 Embedding layer

Geometric heterogeneous graph. Structures are represented as heterogeneous geometric graphs with nodes representing ligand (heavy) atoms, receptor residues (located in the position of the α-carbon atom) and receptor (heavy) atoms (only for the confidence model). Because of the high number of nodes involved it is necessary for the graph to be sparsely connected for runtime and memory constraints. Moreover, sparsity can act as a useful inductive bias for the model, however, it is critical, for the model to find the right pose, that nodes that might have a strong interaction in the final pose to be connected during the diffusion process. Therefore, to build the radius graph, we connect nodes using cutoffs that are dependent on the types of nodes they are connecting:

1. Ligand atoms-ligand atoms, receptor atoms-receptor atoms, and ligand atoms-receptor atoms interactions all use a cutoff of 5Å, standard practice for atomic interactions. For the ligand atoms-ligand atoms interactions we also preserve the covalent bonds as separate edges with some initial embedding representing the bond type (single, double, triple and aromatic). For receptor atoms-receptor atoms interactions, we limit at 8 the maximum number of neighbors of each atom. Note that the ligand atoms-receptor atoms only appear in the confidence model where the final structure is already set.

2. Receptor residues-receptor residues use a cutoff of 15 Å with 24 as the maximum number of neighbors for each residue.

3. Receptor residues-ligand atoms use a cutoff of $20 + 3 * \sigma_{tr}$ Å where σ_{tr} represents the current standard deviation of the diffusion translational noise present in each dimension (zero for the confidence model). Intuitively this guarantees that with high probability any of the ligand and receptors that will be interacting in the final pose the diffusion model converges to are connected in the message passing at every step.

4. Finally, receptor residues are connected to the receptor atoms that form the corresponding amino-acid.

Node and edge featurization. For the receptor residues, we use the residue type as a feature as well as a language model embedding obtained from ESM2 [22]. The ligand atoms have the following features: atomic number; chirality; degree; formal charge; implicit valence; the number of connected hydrogens; the number of radical electrons; hybridization type; whether or not it is in an aromatic ring; in how many rings it is; and finally, 6 features for whether or not it is in a ring of size 3, 4, 5, 6, 7, or 8. This are concatenated with sinusoidal embeddings of the diffusion time [23] and, in the case of edges, radial basis embeddings of edge length [24]. These scalar features of each node and edge are then transformed with learnable two-layer MLPs (different for each node and edge type) into a set of scalar features that are used as initial representations by the interaction layers.

Notation Let $(\mathcal{V}, \mathcal{E})$ represent the heterogeneous graph, with $\mathcal{V} = (\mathcal{V}_\ell, \mathcal{V}_r)$ respectively ligand atoms and receptor residues (receptor atoms $\mathcal{V}_a$, present in the confidence model, are for simplicity not included here), and similarly $\mathcal{E} = (\mathcal{E}_{\ell\ell}, \mathcal{E}_{\ell r}, \mathcal{E}_{r\ell}, \mathcal{E}_{rr})$. Let $\mathbf{h}_a$ be the node embeddings (initially only scalar channels) of node a, e_{ab} the edge embeddings of (a, b), and $\mu(r_{ab})$ radial basis embeddings of the edge length. Let σ^2_{tr}, σ^2_{rot}, and σ^2_{tor} represent the variance of the diffusion kernel in each of the three components: translational, rotational and torsional.

D.2 Interaction layers

At each layer, for every pair of nodes in the graph, we construct messages using tensor products of the current node features with the spherical harmonic representations of the edge vector. The weights of this tensor product are computed based on the edge embeddings and the *scalar* features—denoted $\mathbf{h}^0_a$—of the outgoing and incoming nodes. The messages are then aggregated at each node and used to update the current node features. For every node a of type t_a:

$$\mathbf{h}_a \leftarrow \mathbf{h}_a \underset{t\in\{\ell,r\}}{\oplus} \mathrm{BN}^{(t_a,t)}\left(\frac{1}{|\mathcal{N}_a^{(t)}|}\sum_{b\in\mathcal{N}_a^{(t)}} Y(\hat{r}_{ab}) \otimes_{\psi_{ab}} \mathbf{h}_b\right) \tag{15}$$
$$\text{with } \psi_{ab} = \Psi^{(t_a,t)}(e_{ab}, \mathbf{h}_a^0, \mathbf{h}_b^0)$$

Here, t indicates an arbitrary node type, $\mathcal{N}_a^{(t)} = \{b \mid (a,b) \in \mathcal{E}_{t_a t}\}$ the neighbors of a of type t, Y are the spherical harmonics up to $\ell = 2$, and BN the (equivariant) batch normalisation. The orders of the output are restricted to a maximum of $\ell = 1$. All learnable weights are contained in Ψ, a dictionary of MLPs, which uses different sets of weights for different edge types (as an ordered pair so four types for the score model and nine for the confidence) and different rotational orders.

D.3 Output layer

The ligand atom representations after the final interaction layer are used in the output layer to produce the required outputs. This is where the score and confidence architecture differ significantly. On one hand, the score model's output is in the tangent space $T_{\mathbf{r}}\mathbb{T}_3 \oplus T_R SO(3) \oplus T_{\boldsymbol{\theta}} SO(2)^m$. This corresponds to having two $SE(3)$-equivariant output vectors representing the translational and rotational score predictions and m $SE(3)$-invariant output scalars representing the torsional score. For each of these we design final tensor-product convolutions inspired by classical mechanics. On the other hand, the confidence model outputs a single $SE(3)$-invariant scalar representing the confidence score. Below we detail how each of these outputs is generated.

Translational and rotational scores. The translational and rotational score intuitively represent, respectively, the linear acceleration of the center of mass of the ligand and the angular acceleration of the rest of the molecule around the center. Considering the ligand as a rigid object and given a set of forces and masses at each ligand a tensor product convolution between the atoms and the center of mass would be capable of computing the desired quantities. Therefore, for each of the two outputs, we perform a convolution of each of the ligand atoms with the (unweighted) center of mass c.

$$\mathbf{v} \leftarrow \frac{1}{|\mathcal{V}_\ell|}\sum_{a\in\mathcal{V}_\ell} Y(\hat{r}_{ca}) \otimes_{\psi_{ca}} \mathbf{h}_a \tag{16}$$
$$\text{with } \psi_{ca} = \Psi(\mu(r_{ca}), \mathbf{h}_a^0)$$

We restrict the output of $\mathbf{v}$ to a single odd and a single even vectors (for each of the two scores). Since we are using coarse-grained representations of the protein, the score will neither be even nor odd, therefore, we sum the even and odd vector representations of $\mathbf{v}$. Finally, the magnitude (but not direction) of these vectors is adjusted with an MLP taking as input the current magnitude and the sinusouidal embeddings of the diffusion time. Finally, we (revert the normalization) by multiplying the outputs by $1/\sigma_{tr}$ for the translational score and by the expected magnitude of a score in $SO(3)$ with diffusion parameter σ_{rot} (precomputed numerically).

Torsional score. To predict the m $SE(3)$-invariant scalar describing the torsional score, we use a pseudotorque layer similar to that of [5]. This predicts a scalar score $\delta\tau$ for each rotatable bond from the per-node outputs of the atomic convolution layers. For rotatable bond $g = (g_0, g_1)$ and $b \in \mathcal{V}_\ell$, let r_{gb} and $\hat{r}_{gb}$ be the magnitude and direction of the vector connecting the center of bond g and b. We construct a convolutional filter T_g for each bond g from the tensor product of the spherical harmonics with a $\ell = 2$ representation of the bond axis $\hat{r}_g$:[3]

$$T_g(\hat{r}) := Y^2(\hat{r}_g) \otimes Y(\hat{r}) \tag{17}$$

$\otimes$ is the full (i.e., unweighted) tensor product as described in [25], and the second term contains the spherical harmonics up to $\ell = 2$ (as usual). This filter (which contains orders up to $\ell = 3$) is then

[3]Since the parity of the $\ell = 2$ spherical harmonic is even, this representation is indifferent to the choice of bond direction.

used to convolve with the representations of every neighbor on a radius graph:

$$\begin{aligned}
\mathcal{E}_\tau &= \{(g,b) \mid g \text{ a rotatable bond}, b \in \mathcal{V}_\ell\} \\
e_{gb} &= \Upsilon^{(\tau)}(\mu(r_{gb})) \quad \forall (g,b) \in \mathcal{E}_\tau \\
\mathbf{h}_g &= \frac{1}{|\mathcal{N}_g|} \sum_{b \in \mathcal{N}_g} T_g(\hat{r}_{gb}) \otimes_{\gamma_{gb}} \mathbf{h}_b \\
\text{with } \gamma_{gb} &= \Gamma(e_{gb}, \mathbf{h}_b^0, \mathbf{h}_{g_0}^0 + \mathbf{h}_{g_1}^0)
\end{aligned} \tag{18}$$

Here, $\mathcal{N}_g = \{b \mid (g,b) \in \mathcal{E}_\tau\}$ and $\Upsilon^{(\tau)}$ and Γ are MLPs with learnable parameters. Since unlike [5], we use coarse-grained representations the parity also here is neither even nor odd, the irreps in the output are restricted to arrays both even $\mathbf{h}'_g$ and odd $\mathbf{h}''_g$ scalars. Finally, we produce a single scalar prediction for each bond:

$$\delta\tau_g = \Pi(\mathbf{h}'_g + \mathbf{h}''_g) \tag{19}$$

where Π is a two-layer MLP with $\tanh$ nonlinearity and no biases. This is also "denormalised" multiplying by by the expected magnitude of a score in $SO(2)$ with diffusion parameter σ_{tor}.

Confidence output. The single $SE(3)$-invariant scalar representing the confidence score output is instead obtained by concatenating the even and odd final scalar representation of each ligand atom, averaging these feature vectors among the different atoms and finally applying a three layers MLP (with batch normalization).

E Experimental Details

E.1 Experimental Setup

Data. We use the molecular complexes in PDBBind [14] that were extracted from the Protein Data Bank (PDB) [15]. We employ the time-split of PDBBind proposed by [1] with 17k complexes from 2018 or earlier for training/validation and 363 test structures from 2019. This is motivated by the further adoption of the same split [2] and the critical assessment of PDBBind splits by [16] who favor temporal splits over artificial splits based on molecular scaffolds or protein sequence/structure similarity. For completeness we also report the results on protein sequence similarity splits in Appendix F.

Metrics. To evaluate the generated complexes we compute the heavy-atom RMSD between the predicted and the crystal ligand atoms when the protein structures are aligned. To account for permutation symmetries in the ligand, we use the symmetry-corrected RMSD of sPyRMSD [26]. For these RMSD values, we report the percentage of predictions that have an RMSD that is less than 2Å. We choose 2Å since much prior work considers poses with an RMSD less that 2Å as "good" or successful [27, 10, 11]. This is a chemically relevant metric unlike the mean RMSD as detailed in Appendix B since for further downstream analyses such as determining function changes, a prediction is only useful below a certain RMSD error threshold. Less relevant metrics such as the mean RMSD are provided in Appendix F. We report the metrics for the highest ranked prediction as the top-1; top-5 refers to selecting the best performing pose out of the 5 highest ranked predictions, which can be important for many applications where multiple predictions are used for downstream tasks.

Implementation. We use Adam [28] as optimizer for the diffusion and the confidence model. The diffusion model with which we run inference uses the exponential moving average of the weights during training and we update the moving average after every optimization step with a decay factor of 0.999. The batchsize is 16. We run inference with 20 denoising steps on 500 validation complexes every 5 epochs and use the set of weights with the highest percentage of RMSDs less than 2Å as the final diffusion model. We train on four 48GB RTX A6000 GPUs for 400 epochs (around 9 days). For inference only a single GPU is required. Scaling up the model size seems to improve performance and future work could explore whether this trend continues further. For the confidence model uses the validation cross-entropy loss is used for early stopping and training only takes 70 epochs.

E.2 Baselines: implementation, used scripts, and runtime details

For obtaining the runtimes of the different methods we always used 16 CPUs except for GLIDE as explained below. The runtimes do not include any preprocessing time for any of the methods.

For instance, the time that it takes to run P2Rank is not included for TANKBind and P2Rank + SMINA/GNINA since this receptor preparation only needs to be ran once when docking many ligands to the same protein. In applications where different receptor are processed (such as reverse screening), the experienced runtimes for TANKBind and P2Rank + SMINA/GNINA will thus be higher.

We note that for all these baselines we have used the default hyperparameters unless specified differently below. Modifying some of these hyperparameters (for example the scoring method's exhaustiveness) will change the runtime and performance tradeoffs (e.g. if the searching routine is left running for longer then better poses are likely to be found), however, we leave these analysis to future work.

SMINA [12] improves Autodock Vina with a new scoring-function and user friendliness. The default parameters were used with the exception of setting `-num_modes 10`. To define the search box, we use the automatic box creation option around the receptor with the default buffer of 4Å on all 6 sides.

GNINA [11] builds on SMINA by additionally using a learned 3D CNN for scoring. The default parameters were used with the exception of setting `-num_modes 10`. To define the search box, we use the automatic box creation option around the receptor with the default buffer of 4Å on all 6 sides.

QuickVina-W [10] extends the speed-optimized QuickVina 2 [27] for blind docking. We reuse the numbers from [1] which had used the default parameters except for increasing the exhaustiveness to 64.

GLIDE [13] is a strong heavily used commercial docking tool. These methods all use biophysics based scoring-functions. We reuse the numbers from [1] since we do not have a license. As explained by [1], the very high runtime of GLIDE with 1405 seconds per complex is partially explained by the fact that GLIDE only uses a single thread when processing a complex. This fact and the parallelization options of GLIDE are explained here `https://www.schrodinger.com/kb/1165`. With GLIDE, it is possible to start data-parallel processes that compute the docking results for a different complex in parallel. However, in the mentioned link, it is explained that each process also requires a separate software license.

EquiBind [1], we reuse the numbers reported in their paper and generate the predictions that we visualize with their code at `https://github.com/HannesStark/EquiBind`.

TANKBind [2], we use the code associated with the paper at `https://github.com/luwei0917/TankBind`. The runtimes do not include the runtime of P2Rank or any preprocessing steps. In Table 1 we report two runtimes for GPU (0.72/2.5 sec) and CPU (1.95/3.7 sec). The first is the runtime when making only the top-1 prediction and the second is for producing the top-5 predictions. Producing only the top-1 predictions is faster since TANKBind produces distance predictions that need to be converted to coordinates with a gradient descent algorithm and this step only needs to be run once for the top-1 prediction while it needs to be run 5 times for producing 5 outputs. To obtain our runtimes we run the forward pass of TANKBind on an RTX A6000 GPU or on an Intel Xeon Gold 6230 CPU with the default batchsize of 5 that is used in their GitHub repository. To compute the time the distances-to-coordinates conversion step takes, we parallelizes the computation across 16 processes which we also run on an Intel Xeon Gold 6230 CPU. This way we obtain 0.44 seconds runtime for the conversion step of the top-1 prediction (averaged over the 363 complexes of the testset).

P2Rank [29], is a tool that predicts multiple binding pockets and ranks them. We use it for running TANKBind and P2Rank + SMINA/GNINA. We download the program from `https://github.com/rdk/p2rank` and run it with its default parameters.

EquiBind + SMINA/GNINA [1], the bounding box in which GNINA/SMINA searches for binding poses is constructed around the prediction of EquiBind with the `-autobox_ligand` option of GNINA/SMINA. EquiBind is thus used to find the binding pocket and SMINA/GNINA to find the exact final binding pose. We use `-autobox_add 10` to add an additional 10Å on all 6 sides of the bounding box following [1].

P2Rank + SMINA/GNINA. The bounding box in which GNINA/SMINA searches for binding poses is constructed around the pocket center that P2Rank predicts as the most likely binding pocket. P2Rank is thus used to find the binding pocket and SMINA/GNINA to find the exact final binding pose. The diameter of the search box is the diameter of a ligand conformer generated by RDKit with an additional 10Å on all 6 sides of the bounding box.

F Additional Results

F.1 Physically plausible predictions

Table 2: **Steric clashes.** Percentage of test complexes for which the predictions of the different methods exhibit steric clashes. Biophysics scoring-function based methods never produced steric clashes.

Method	Top-1 % steric clashes	Top-5 % steric clashes
EQUIBIND	26	-
TANKBIND	7.2	4.4
DIFFDOCK	**4.4**	**0.3**

Due to the averaging phenomenon of regression based methods such as TANKBind and EquiBind, they make predictions at the mean of the distribution. If aleatoric uncertainty is present, such as in case of symmetric complexes, this leads to predicting the ligand to be at an un-physical state in the middle of the possible binding pockets as visualized in Figure 7. The Figure also illustrates how DIFFDOCK does not suffer from this issue and is able to accurately sample from the modes.

In the scenario when epistemic uncertainty about the correct ligand conformation is present, this often results in "squashed-up" predictions of the regression based methods as visualized in Figure 3. If there is uncertainty about the correct conformer, the square error minimizing option is to put all atoms close to the mean.

These averaging phenomena in the presence of either aleatoric or epistemic uncertainty cause the regression based methods to often generate steric clashes and self intersections. To investigate this quantitatively, we determine the fraction of test complexes for which the methods exhibit steric clashes. We define a ligand as exhibiting a steric clash, if one of its heavy atoms is within 0.4Å of a heavy receptor atom. This cutoff is used by protein quality assessment tools and in previous literature [30]. Table 2 shows that DIFFDOCK, as a generative model, produces fewer steric clashes than the regression based baselines. We generally observe no unphysical predictions from DIFFDOCK unlike the self intersections that, e.g., TANKBind produces (Figure 3) or its incorrect local structures (Figure 4).

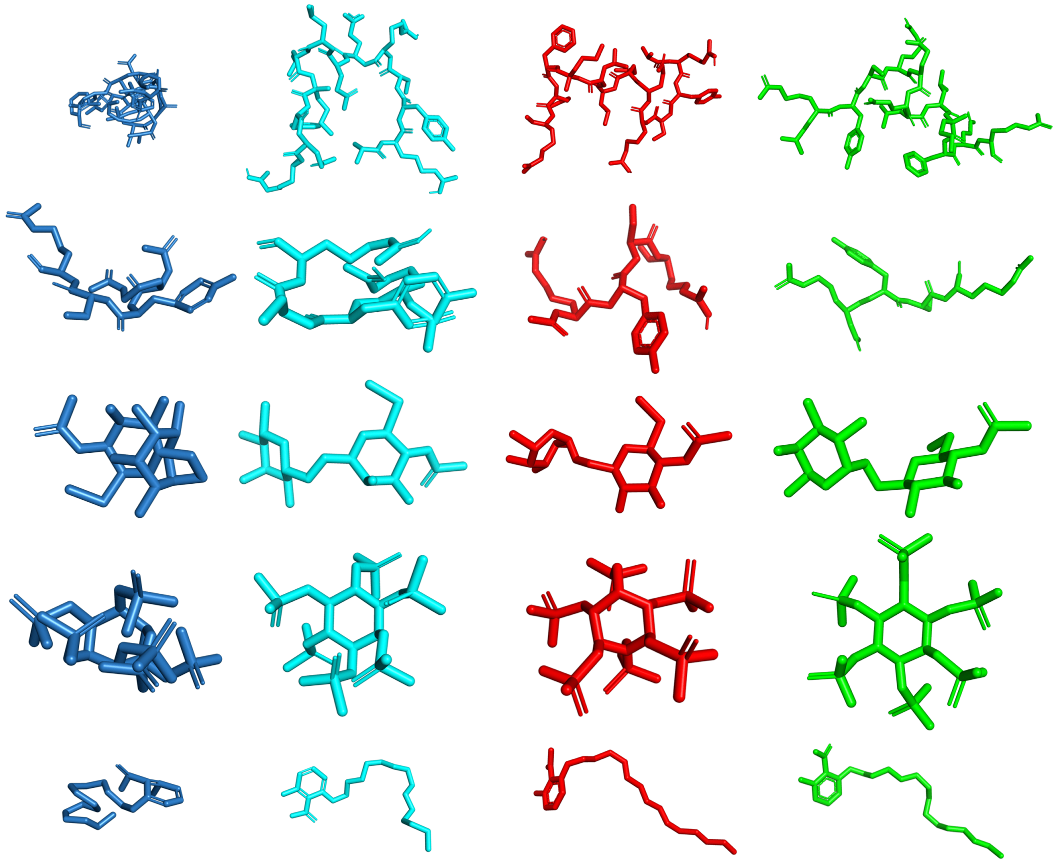

Figure 3: **Ligand self-intersections.** TANKBind (blue), EquiBind (cyan), DiffDock (red), and crystal structure (green). Due to the averaging phenomenon that occurs when epistemic uncertainty is present, the regression based deep learning models tend to produce ligands with atoms that are close together leading to self-intersections. DiffDock, as a generative model, does not suffer from this averaging phenomenon and we never found a self-intersection in any of the investigated results of DiffDock.

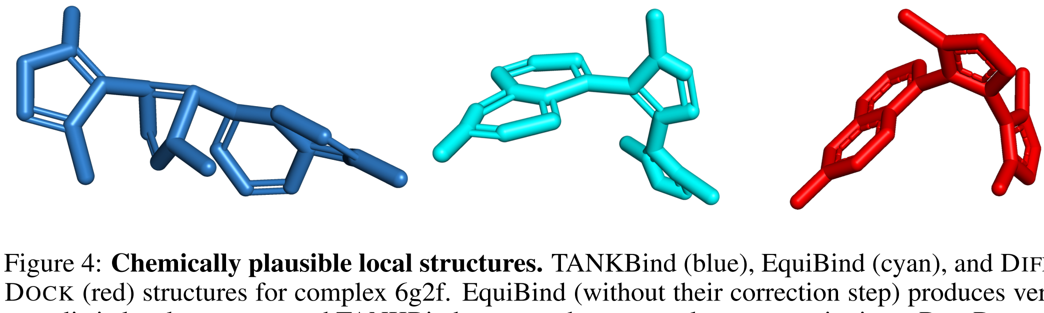

Figure 4: **Chemically plausible local structures.** TANKBind (blue), EquiBind (cyan), and DiffDock (red) structures for complex 6g2f. EquiBind (without their correction step) produces very unrealistic local structures and TANKBind, e.g., produces non-planar aromatic rings. DiffDock's local structures are the realistic local structures of RDKit.

F.2 Further results and selective accuracy

Figure 5-*left* shows the proportion of RMSDs below an arbitrary threshold ϵ with DiffDock exceeding previous methods for almost every possible ϵ. With the exception of very small ϵ <1Å where GLIDE performs better than the other methods because it is the only one directly including a final energy relaxation of the local structures inside the method, but this final relaxation could be added to any of the other methods.

As the top-1 results show, DiffDock's confidence model is very accurate in ranking the sampled poses for a given complex and picking the best one. We also investigate the *selective accuracy* of this score across *different* complexes, by evaluating how DiffDock's accuracy increases if it only makes predictions when the confidence is above a certain threshold, known as *selective prediction*.

In Figure 5 we plot the "success" rate as we decrease the percentage of complexes for which we make predictions (*selective accuracy*), corresponding to increasing confidence threshold. When only making predictions for the top one-third of complexes in terms of model confidence, the (RMSD $<$ 2Å)-rate improves from 37% to 85%. Additionally, there is a high Spearman correlation of 0.74 between DiffDock's confidence and the RMSD. Evidently, the confidence score is informative for judging the quality of DiffDock's top-ranked sampled pose and provides a highly valuable confidence measure for practitioners, a critical factor for many downstream applications.

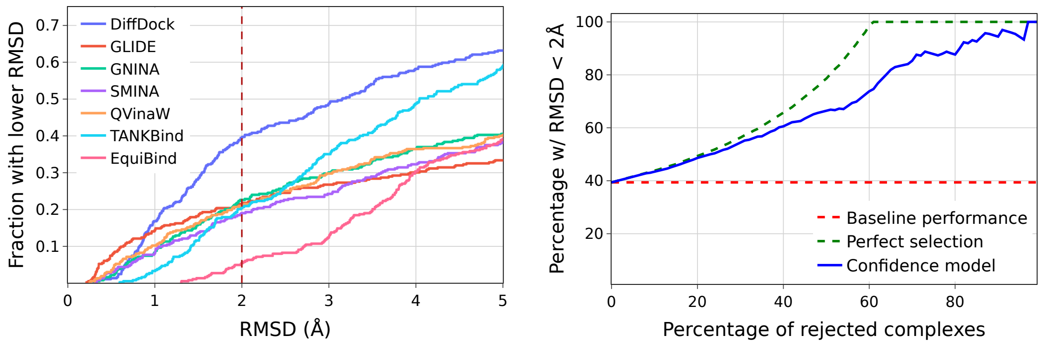

Figure 5: **Left:** cumulative density histogram of the methods' RMSD. **Right:** DiffDock's selective accuracy, the percentage of predictions with RMSD below 2Å when only making predictions for the percentage of the dataset on the x-axis for which DiffDock is the most confident.

F.3 Visualizations

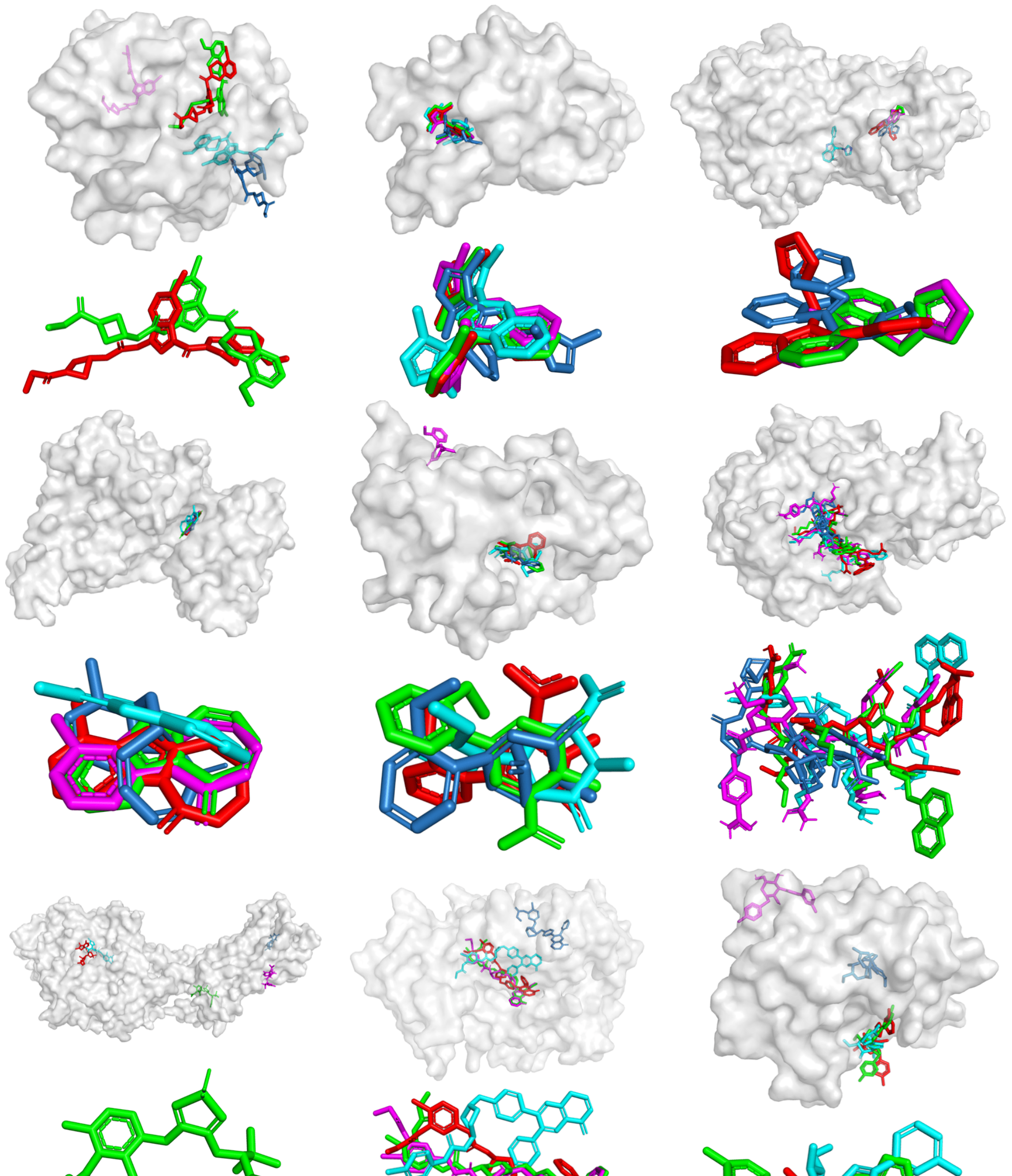
Figure 6: **Randomly picked examples.** The predictions of TANKBind (blue), EquiBind (cyan), GNINA (magenta), DiffDock (red), and crystal structure (green). Shown are the predictions once with the protein and without it below. The complexes were chosen with a random number generator from the test set. TANKBind often produces self intersections (examples at the top-right; middle-middle; middle-right; bottom-right). DiffDock and GNINA sometimes almost perfectly predict the bound structure (e.g., top-middle). The complexes in reading order are: 6p8y, 6mo8, 6pya, 6t6a, 6e30, 6hld, 6qzh, 6hhg, 6qln.

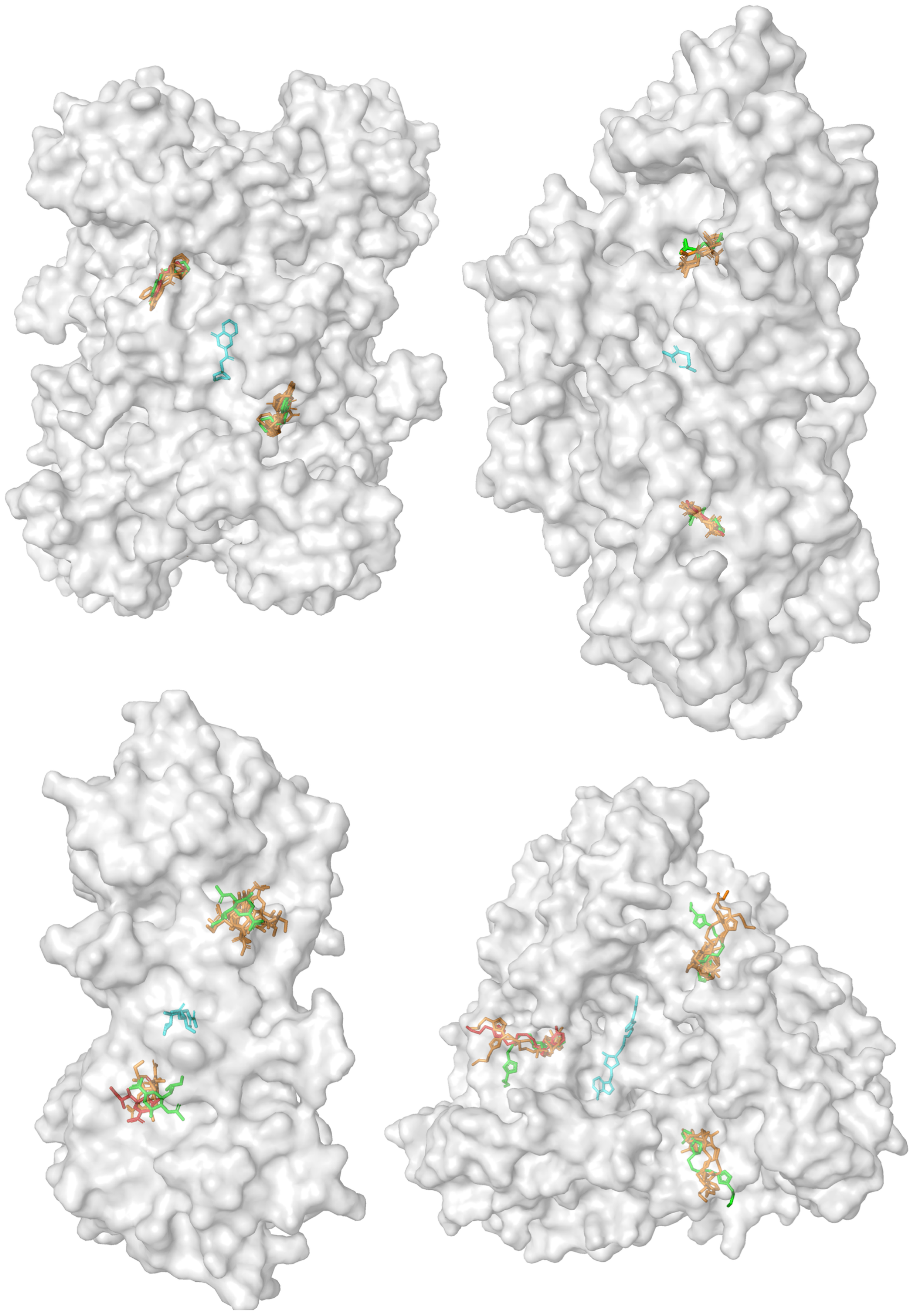

Figure 7: **Symmetric complexes and multiple modes.** EquiBind (cyan), DIFFDOCK highest confidence sample (red), all other DIFFDOCK samples (orange), and the cystal structure (green). We see that, since it is a generative model, DIFFDOCK is able to produce multiple correct modes and to sample around them. Meanwhile, as a regression based model, EquiBind is only able to predict a structure at the mean of the modes. The complexes are unseen during training. The PDB IDs in reading order: 6agt, 6gdy, 6ckl, 6dz3.

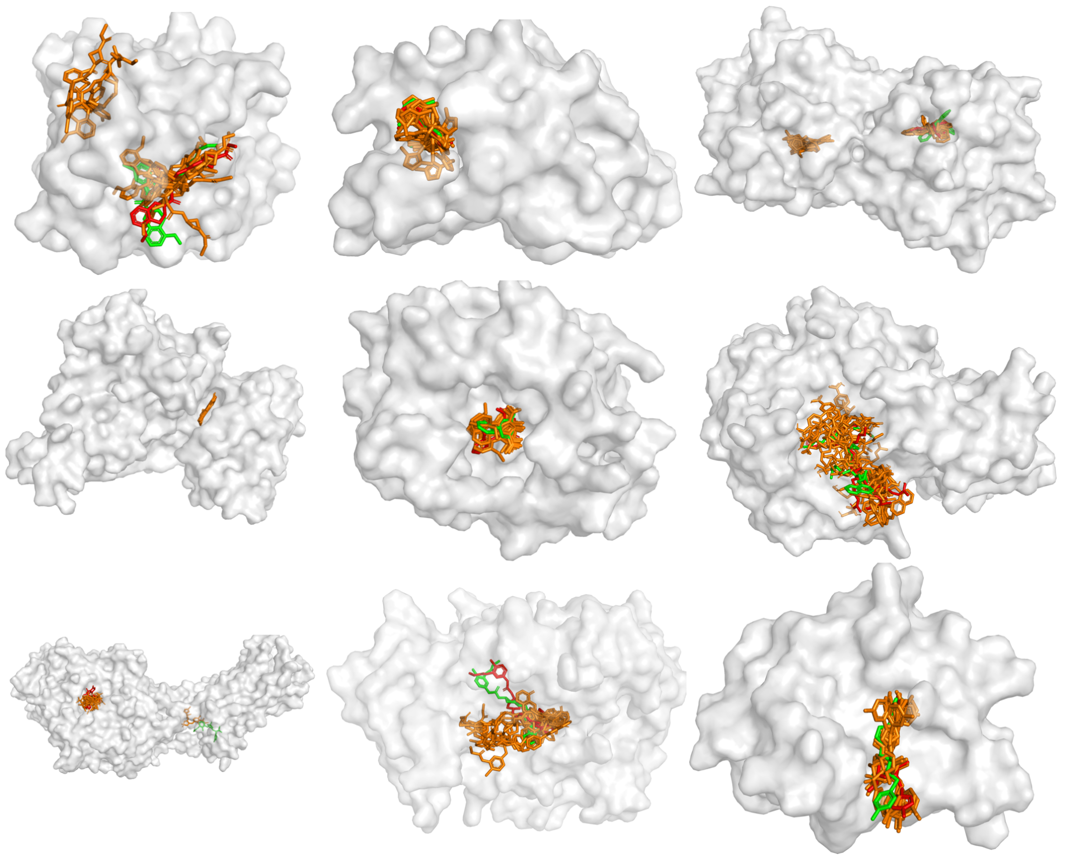

Figure 8: **Generated samples for randomly picked examples.** Generated samples after the reverse diffusion. Shown are DIFFDOCK highest confidence sample (red), all other DIFFDOCK samples (orange), and the cystal structure (green). The complexes were chosen with a random number generator from the test set. The complexes in reading order are: 6p8y, 6mo8, 6pya, 6t6a, 6e30, 6hld, 6qzh, 6hhg, 6qln.

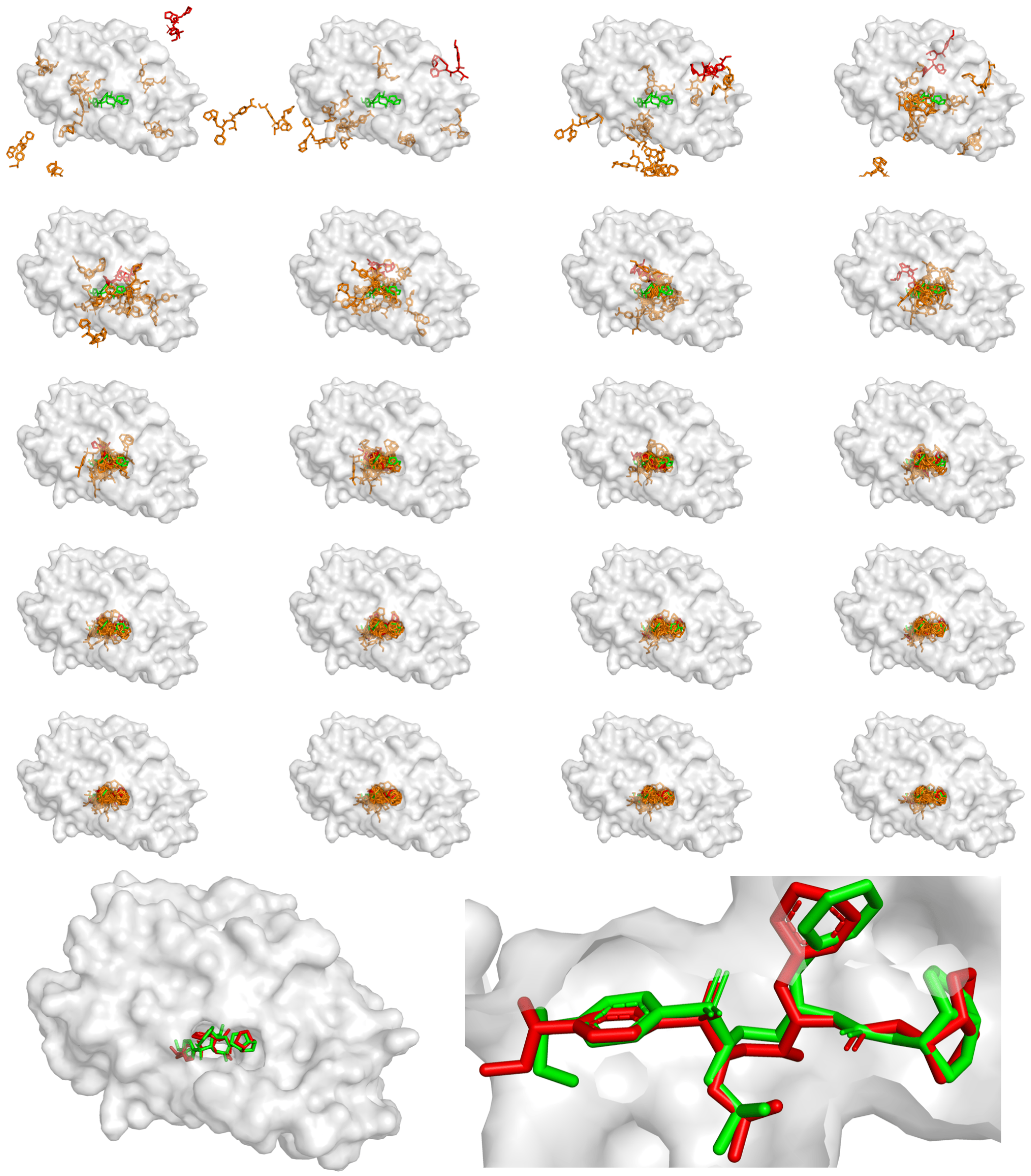

Figure 9: **Reverse Diffusion.** Reverse diffusion of a randomly picked complex from the test set. Shown are DIFFDOCK highest confidence sample (red), all other DIFFDOCK samples (orange), and the cystal structure (green). Shown are the 20 steps of reverse diffusion process (in reading order) of DIFFDOCK for the complex 6oxx.

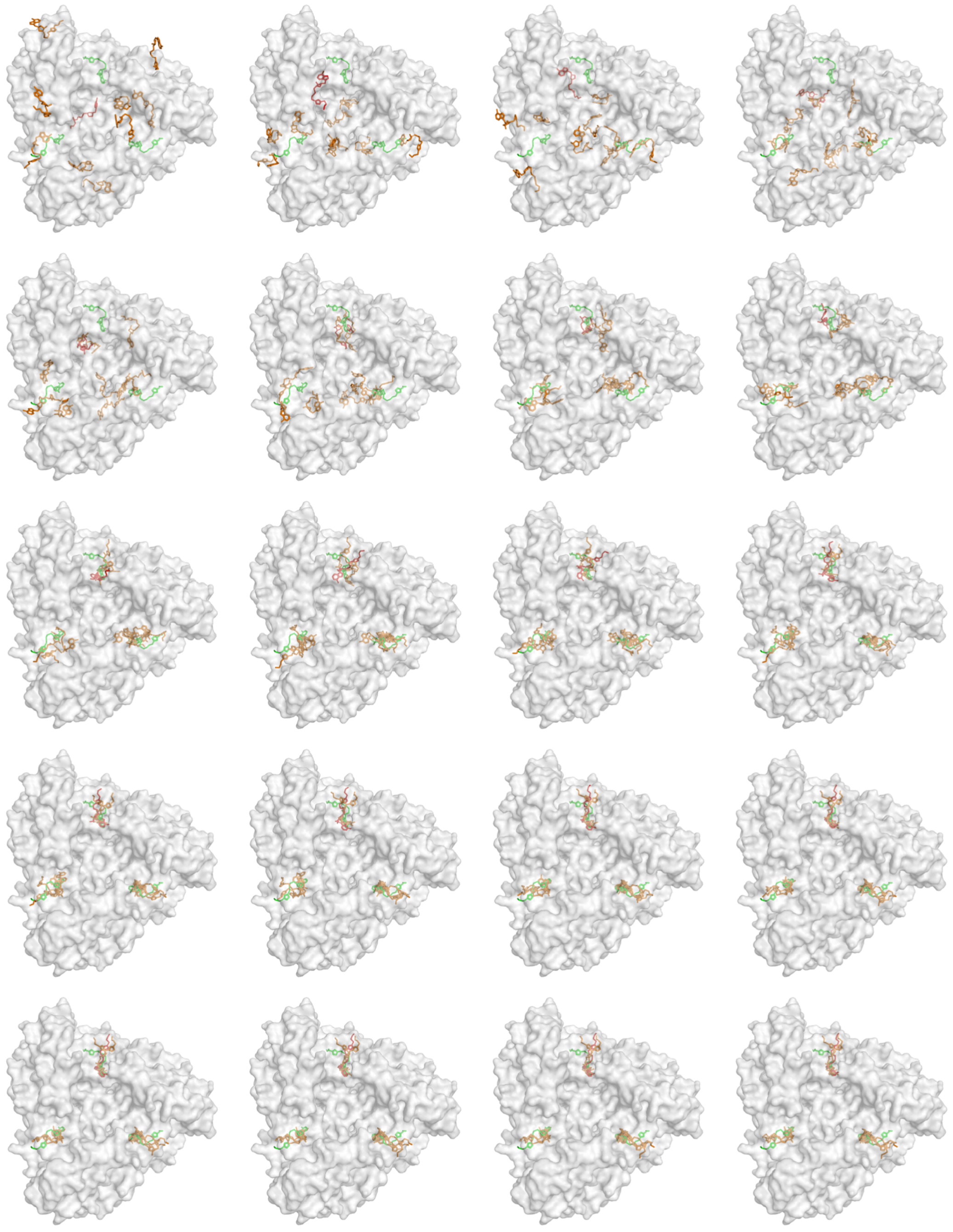

Figure 10: **Reverse Diffusion.** DIFFDOCK highest confidence sample (red), all other DIFFDOCK samples (orange), and the cystal structure (green). Shown are the 20 steps of reverse diffusion process (in reading order) of DIFFDOCK for the complex 6dz3 which was not seen during training.